# CS598 Deep Learning for Healthcare - Final Reproducibility Project - SurvTRACE: Transformers for Survival Analysis with Competing Events

**John Hanratty and Rodrigo Rodrigues**
{johnh7, rar5}@illinois.edu

Group ID: 42, Paper ID: 313 (Hard)
Presentation link: https://www.youtube.com/watch?v=IAoQO6pEZWI
Code link: https://github.com/jhanratty/CS598DLH/

## 1 Introduction

Survival analysis is a collection of statistical procedures for data analysis where the outcome variable of interest is the time until an event occurs. Because of censoring — the non-observation of the event of interest after a period of follow-up — a proportion of the survival times of interest will often be unknown (Clark et al., 2003). The name implies a prediction of death but survival analysis can predict other medical conditions, mechanical failures, or stock market events (Wikimedia, 2021).

The SurvTRACE paper (Wang and Sun, 2021) proposes the use of a transformer architecture for survival analysis modeling. The authors hypothesize that this innovation can provide the following benefits:

1. Outperform methods that ignore selection bias, especially on the analysis of rare events
2. Automate learning of high-order interactions between covariates through attention
3. Create a shared representation for data as a multi-task learning (MTL) backbone for multiple prediction tasks

Our project focused first on implementing a simple version of the SurvTRACE competing event architecture, verifying the results stated in the paper, and experimenting with improvements, hyperparameters and extensions. In addition, the team extended the code support a unified model for both single and competing event applications. (The demo code supported these as separate models.) The paper was chosen with the desire to learn more about transformer and attention deep learning models

## 2 Scope of reproducibility

Our team was able to reproduce the model using open source libraries and achieve similar metrics when compared to the results presented in the paper. The demo code provided with the paper datasets that made it possible to compare results directly.

### 2.1 Addressed claims from the original paper

- **Automatic feature engineering with attentive encoders.**

  The paper describes a method for 'automatically feature engineering' using attention encoding that learns high order interaction between covariates. The claim is that the models using this method produce comparable or better results as previous methods that require more manual feature engineering.

- **Multi-task learning with a shared backbone**

  The paper proposes shared backbone architecture that learns to predict multiple events (e.g. Heart Disease and Breast Cancer).. Other approaches predict only a single event.. The claim is that this strengthens prediction accuracy for all events.

- **Debiasing competing events analysis using counter- factual learning**

  The paper proposes the use of inverse propensity score(Little and Rubin, 2019) based logistic hazard loss. IPS-based Loss as is an Unbiased Estimate of True Loss, demonstrated at the end of the paper in appendix A.

## 3 Methodology

The model was developed in the format of a clear and simple Jupyter Notebook, with the intent of

making it easy to understand and run, the code will automatically detect a CUDA compatible GPU making use of it if available.

The new, re-produced model is based on a number of Survival Analysis tool kits that include basic models, datasets, evaluation metrics, and other utilities:

- Pycox - Time-to-event prediction with PyTorch (Kvamme, 2002).
- Scikit-survival - module for survival analysis built on top of scikit-learn (Pölsterl, 2020).
- Pytorch - open-source machine learning framework (pyt, 2022).
- TorchTuples - open-source python package for handling model training data in form of tuples (Kvamme, 2022).
- Hugging Bear - transformer software kit (Team, 2022).

### 3.1 Model descriptions

The model's architecture is captured in the diagram in Figure 1 and can be represented as a 3-stage structure.

- **Stage1**

  The input features consist of concatenated embeddings from categorical and numerical data features. The numerical values are normalized and the categorical features are assigned unique embedding values. These embedded values are concatenated and fed into the encoder stage. This stage was implemented using torch, sklearn and pycox utilities. Pycox provides an extensive collection of survival analysis examples, datasets and utilities.

  The target values (Y) from the data set include event_breast, event_heart, and duration. The pycox DiscretizeUnknownC() function was used to process this data. The duration value is converted from months to 4 discrete periods (1-17, 18-34, 35-58, 59-121 months) for when the event occurred. The events are converted value of 1 if they happen and 0 if not. The duration's proportion of the assigned discrete period as is added as a column to support the loss function for training.

- **Stage2**

  A transformer encoder stage with multi-headed, self-attention automatically learns the "combinatorial interactions" between the embeddings from the first stage. The output is a shared 'backplane' representation of the input features that feed multiple feed forward networks that provide hazard predictions for 'competing events' (e.g. heart disease, breast cancer). This transformer stage was implemented using the Hugging Bear BART implementation.

- **Stage3**

  The final stage uses two feedforward networks with softplus outputs to predict the hazard ratio for four discrete durations for breast cancer and heart disease.

### 3.2 Data descriptions

The data set is extracted from the SEER - Surveillance, Epidemiology, and End Results Program (National Cancer Institute, 2022) data. The raw fields were selected and downloaded using a Python script created by the paper authors. The data set included 475,745 entries and 20 features (shown in Figure 2).

The following data sets are part of the pycox offering and were used for testing the code in a single event configuration.

- METABRIC - The Molecular Taxonomy of Breast Cancer International Consortium (Kvamme, 2002).
- SUPPORT - Study to Understand Prognoses Preferences Outcomes and Risks of Treatment (Kvamme, 2002).

### 3.3 Hyperparameters

Hyperparameters were grid tested for the optimal configuration. The model was training using the following list of Hyperparameters:

Encoder

```
# embedding size
'hidden_size': 16,
# intermediate layer size in
# transformer layer
'intermediate_size': 64,
# num of transformers
'num_hidden_layers': 2,
# num of attention heads in
# transformer layer
'num_attention_heads': 2,
'hidden_dropout_prob': 0.0,
'attention_probs_dropout_prob': 0.1,
```

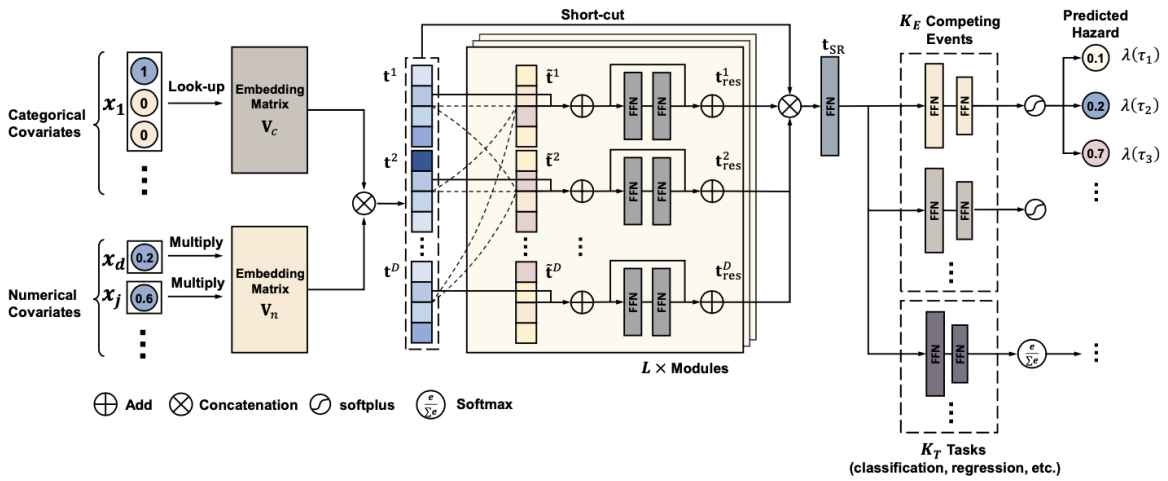


Figure 1: Model Diagram.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 476746 entries, 0 to 476745
Data columns (total 21 columns):
 #   Column                                                  Non-Null Count   Dtype
---  ------                                                  --------------   -----
 0   Sex                                                     476746 non-null  int64
 1   Year of diagnosis                                       476746 non-null  int64
 2   Race recode (W, B, AI, API)                             476746 non-null  int64
 3   Histologic Type ICD-O-3                                 476746 non-null  int64
 4   Laterality                                              476746 non-null  int64
 5   Sequence number                                         476746 non-null  int64
 6   ER Status Recode Breast Cancer (1990+)                  476746 non-null  int64
 7   PR Status Recode Breast Cancer (1990+)                  476746 non-null  int64
 8   Summary stage 2000 (1998-2017)                          476746 non-null  int64
 9   RX Summ--Surg Prim Site (1998+)                         476746 non-null  int64
 10  Reason no cancer-directed surgery                       476746 non-null  int64
 11  First malignant primary indicator                       476746 non-null  int64
 12  Diagnostic Confirmation                                 476746 non-null  int64
 13  Median household income inflation adj to 2019           476746 non-null  int64
 14  Regional nodes examined (1988+)                         476746 non-null  float64
 15  CS tumor size (2004-2015)                               476746 non-null  float64
 16  Total number of benign/borderline tumors for patient    476746 non-null  float64
 17  Total number of in situ/malignant tumors for patient    476746 non-null  float64
 18  duration                                                476746 non-null  int64
 19  event_heart                                             476746 non-null  float64
 20  event_breast                                            476746 non-null  float64
dtypes: float64(6), int64(15)
memory usage: 76.4 MB
```

Figure 2: SEER Dataset.

FF Networks

```
intermediate_size = 64
out_feature = 4
```

Training

```
EPOCHS = 20
BATCH_SIZE = 1280
#learning rate default=1e-3
LR = 1e-3
#weight decay default=1e-4
WT_DECAY = 1e-4
```

### 3.4 Implementation

The architecture for the model developed is outlined above in section 3.1 in Figure 1.

The SurvTRACE demo source code provided by the authors was a useful reference for development but was too complex for our purposes due to structure, flexibility, and many features outside the scope of our project. The demo also provided a useful sanity check to verify components as we developed them. A subset of the functionality was independently developed on top of the open-source pycox (Kvamme, 2002), and scikit-survival (Pölsterl, 2020) tool kits. Transformer code from Hugging Bear (Team, 2022) was also used as the basis for the encoder modules. Pycox provided useful data sets, code examples for a variety of survival analysis model types and many useful utilities for implementing our model. The resulting source code base is considerably smaller and easier to adapt for this project.

The team focused development on a multi-task competing event model that produces similar results as the demo model. This model was further extended into a unified model for both single and competing event prediction. The demo code used two separate models.

### 3.5 Computational requirements

The team used Google Colab to implement and test the models. The model was designed to take advantage of GPUs when available.

The GPU setup was able to perform training with 100 epochs in a few seconds vs. several minutes for a CPU set up. Computational resources were not an issue for completing the project.

## 4 Results

The model performs comparably to the results presented in the paper for competing event (multi-task) prediction (Figure 3) and single event prediction. The model had a slightly lower performance than the paper for the heart disease events in the competing event scenario. Further study is required but this might be related to rarity of heart disease compared to breast cancer in the data set. The model compares favorably against other survial analysis models (Figure 5). The sksurv concordance_index_ipcw (closer to 1 is better) and brier_score (Pölsterl, 2020) (closer to 0 is better) are used for comparison (Figure 4).

With GPU support, the model can be trained in a few seconds, making grid type parameter analysis feasible.

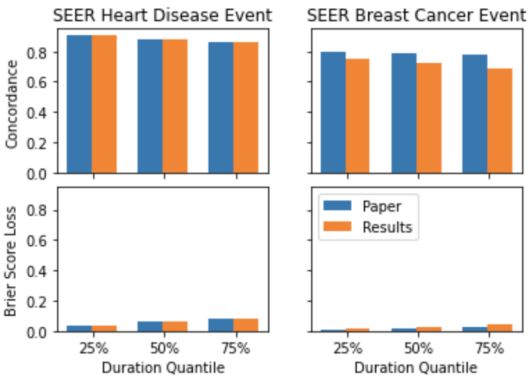


Figure 3: Competing Event Model Results vs. SurvTRACE Paper.

| | Results-Breast \| | | Results-Heart \| | | SurvTRACE-Breat \| | | SurvTRACe-Heart \| | |
|---|---|---|---|---|---|---|---|---|
| | Concordance | Brier | Concordance | Brier | Concordance | Brier | Concordance | Brier |
| 25% | 0.9048 | 0.0356 | 0.7522 | 0.0127 | 0.9040 | 0.0350 | 0.7970 | 0.0080 |
| 50% | 0.8821 | 0.0607 | 0.7226 | 0.0277 | 0.8830 | 0.0600 | 0.7880 | 0.0160 |
| 75% | 0.8639 | 0.0826 | 0.6906 | 0.0443 | 0.8660 | 0.0820 | 0.7750 | 0.0280 |

Figure 4: Competing Event Model Results vs. SurvTRACE Paper.

### 4.1 Automatic feature engineering with attentive en- coders

The paper describes a method for 'automatically feature engineering' using attention encoding that learns high order interaction between covariates. The claim is that the models using this method produce comparable or better results as previous methods that require more manual feature engineering.

### 4.2 Multi-task learning with a shared backbone

The paper proposes shared backbone architecture that learns to predict multiple events (e.g. Heart Disease and Breast Cancer).. Other approaches

predict only a single event.. The claim is that this strengthens prediction accuracy for all events.

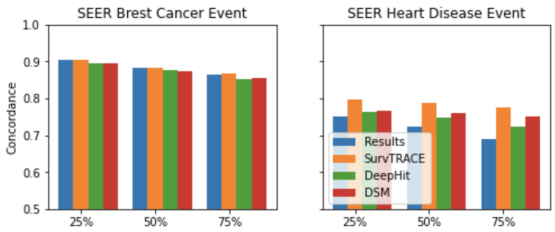


Figure 5: Muli-task Learning Compared to Other Architectures.

| | Testing \| | | SurvTRACE \| | | DeepHit \| | | DSM \| | |
|---|---|---|---|---|---|---|---|---|
| | Breast | Heart | Breast | Heart | Breast | Heart | Breast | Heart |
| **25%** | 0.9032 | 0.7449 | 0.9040 | 0.7970 | 0.8960 | 0.7630 | 0.8950 | 0.7650 |
| **50%** | 0.8818 | 0.7145 | 0.8830 | 0.7880 | 0.8750 | 0.7480 | 0.8730 | 0.7610 |
| **75%** | 0.8630 | 0.6834 | 0.8660 | 0.7750 | 0.8520 | 0.7240 | 0.8560 | 0.7500 |

Figure 6: Muli-task Learning Compared to Other Architectures.

### 4.3 Debiasing competing events analysis using counter- factual learning

The papers presents a mathematical demonstration of the property for the proposed loss function, no further analysis was necessary on this topic.

### 4.4 Additional results not present in the original paper

The demo model provided the SEER dataset that we used for testing and comparison between the original demo and the recreated version. We also tested both models using the PyCox version of METABRIC and SUPPORT datasets, and the models performed as expected in those datasets.

## 5 Discussion

### 5.1 What was easy

The SurvTRACE authors provided demo source codes and ready access to data sets. This provided an excellent quick start as many teams had to scramble to acquire data sets to get started. The source code, although rather confusing provided, a good reference point for functionality and testing. The pycox provided datasets enabled code development while we applied for access to SEER.

### 5.2 What was difficult

The project required a large learning curve for survival analysis data, algorithms, approaches, and utilities.

### 5.3 Recommendations for reproducibility

Pycox (Kvamme, 2002), and scikit-survival (Pölsterl, 2020) tool kits provide invaluable data sets, tutorial examples, utilities and references for survival analysis. Anyone new to the subject can learn about the subject and find useful utilities and code to get started.

## 6 Communication with original authors

We met via Zoom with Professor Jimeng Sun, who co-authored the paper. He suggested some possible additional work but said that reproducing the results was a large task and more than sufficient to successfully complete the assignment.

## References

2022. Pytorch - end-to-end machine learning framework.

T G Clark, M J Bradburn, S B Love, and D G Altman. 2003. Survival analysis part i: Basic concepts and first analyses. *British Journal of Cancer*, 89(2):232–238.

Haavard Kvamme. 2002. Github - havakv/pycox: Survival analysis with pytorch.

Haavard Kvamme. 2022. Github - havakv/torchtuples: Training neural networks in pytorch.

Roderick Little and Donald Rubin. 2019. *Statistical Analysis with Missing Data, Third Edition*. Wiley.

National Cancer Institute. 2022. Surviellence, epidemiology, and end result (seer) data set.

Sebastian Pölsterl. 2020. scikit-survival: A library for time-to-event analysis built on top of scikit-learn. *Journal of Machine Learning Research*, 21(212):1–6.

The Hugging Face Team. 2022. Source code for transformers.models.bart.modeling_bart.

Zifeng Wang and Jimeng Sun. 2021. Survtrace: Transformers for survival analysis with competing events.

Wikimedia. 2021. Survival analysis.